

Търсене на елемент

Задачата за „търсене“ на елементи в даден масив може да бъде определена по следния начин:

Да се намери онзи елемент от даден масив, който е в релация ρ спрямо отнапред зададен елемент a^* . Релацията ρ може да бъде една от следните шест: $<$, $\leq$, $=$, $\neq$, $\geq$, $>$. Понякога се налага да бъдат търсени всичките елементи, които удовлетворяват зададеното условие, но описаните по-долу алгоритми разглеждат първия случай. Разбира се, необходими са малки промени в описанието на дадените алгоритми, за да бъдат валидни и във втория случай.

1. Алгоритъм за последователно търсене

Нека A е едномерен масив с n елемента. Започвайки от първия елемент a_1 последователно се преглеждат следващите елементи от масива A до намирането на търсения или до изчерпването на всички елементи, без да бъде намерен търсеният документ. Очевидно процедурата е много проста за изпълнение но тя „заплаща“ с малката си скорост. Методът на последователното търсене може да бъде ефективен само за малки стойности на n .

Алгоритъм S

S1. [Начален номер на проверявания елемент.] $i:=1$

S2. [Сравнение.] Ако A_i е в релация ρ спрямо a^* , изпълнете S5

S3. [Определяне номера на следващия елемент.] $i:=i+1$

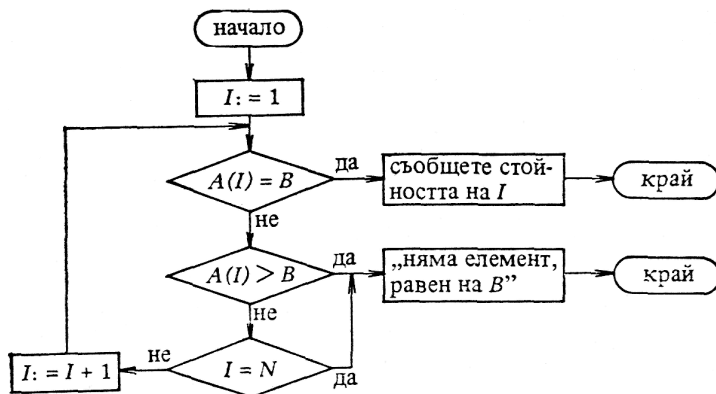
S4. [Има ли още елементи?] Ако $1 \leq n$, изпълнете S2

S5. [Край.] Прекратете търсенето.

Търсенето се извършва последователно, като се започне от първия елемент до намиране на елемент, равен на B или до

намиране на индекс j при който $A_j > B$.

Блок-схема на алгоритъма:



2. Търсене на елемент в подреден масив (двоично търсене)

Даден е масив A с N числени елемента, $N > 1$, подредени по големина в растящ ред. Дадено е и числото b . Да се провери има ли сред елементите на A равен на b . Ако има, да се съобщи неговият индекс, ако не - да се издаде подходящо съобщение.

Идея на алгоритъма: масивът се разделя на две части и се преценява в коя част би могъл да бъде елементът b . С тази част се постъпва по същия начин, както с целия масив. Така се действа, докато се открие елементът b или се стигне до два елемента, за които е в сила $A_j < b$ и $b < A_{j+1}$, което означава, че b не се съдържа в масива A .

Ще предпологаме, че релацията за сравнение е равенство и че масивът A е предварително подреден по възходящ ред на елементите си, т. е. $a_1 \leq a_2 \leq a_3 \leq \dots \leq a_n$.

Ако търсеният елемент е b , веднага може да се определи в коя от двете половини на масива A той не се намира. За целта е

досатъчно да се сравнят b и a_k , където $k = \left\lceil \frac{n}{2} \right\rceil$. По този начин броят на елементите, в които се извършва търсенето, след едно сравнение намалява наполовина. С подходящата част от двете части на масива се постъпва по същия начин, т.е. разделя се отново на две равни части и т.н. Това продължава, докато се намери търсеният елемент или се установи, че такъв елемент в масива няма. Точното описание е следното.

Алгоритъм В

В1. [Начални граници на масива.] $i := 1, j := n$

В2. [Пресмятане „средата“ на масива.] $k = \left\lceil \frac{i+j}{2} \right\rceil$

В3. [Сравнение.] Ако $a_k = b$, изпълнете В8

В4. [Сравнение.] Ако $a_k < b$, изпълнете В6

В5. [Нова дясна граница на масива.] $j := k-1$, изпълнете В7

В6. [Нова лява граница на масива.] $i := k+1$

В7. [Проверка за край.] Ако $i \leq j$, изпълнете В2

В8. [Край.] Прекратете търсенето.

Пример $A = (3, 5, 6, 6, 10, 15, 16, 17, 90, 99)$, $b = 15$, $n = 10$

Изпълнявана точка	Извършено действие	i	j	l
В1	$i := 1$	1	10	
	$j := 10 \rightarrow \text{В2}$			
В2	$k = \left\lceil \frac{1+10}{2} \right\rceil$ $\rightarrow \text{В3}$			5
В3	$a_5 = 15? -$			

	не $\rightarrow$ B4			
B4	$a_5 < 15?$ - да $\rightarrow$ B6			
B6	$i := 6 \rightarrow$ B7	6		
B7	$i \leq j?$ - да $\rightarrow$ B2			
B2	$k := \left\lceil \frac{6+10}{2} \right\rceil$ $\rightarrow$ B3			8
B3	$a_l = 15?$ - не $\rightarrow$ B4			
B4	$a_l < 15?$ - не $\rightarrow$ B5 $a, < 15?$ — не- $> B5$			
B5	$j := 7 \rightarrow$ B7		7	
B7	$i \leq j?$ - да $\rightarrow$ B2			
B2	$k := \left\lceil \frac{6+2}{2} \right\rceil \rightarrow$ B3			6
B3	$a_k = 15?$ - да $\rightarrow$ B8			
B8	край			

Разгледаният масив в примера е много малък, за да бъде добре осъзната голямата скорост на търсене по алгоритъм В. Това по-добре може да се види, ако се пресметне максималният брой сравнения, които се извършват при двоичното търсене. Нека е даден масив с n елемента и l е такова цяло положително число, че:

$$2^{l-1} \leq n < 2^l$$

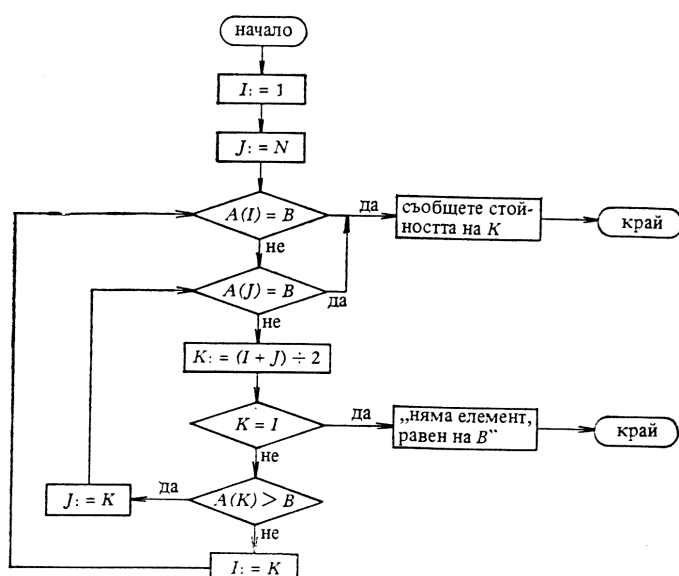
Преди да се извърши първото сравнение, броят на елементите, измежду които е възможно да се намира търсеният елемент, е равен на n . След първото сравнение този брой е вече $\left\lceil \frac{n}{2^1} \right\rceil$, след второто е $\left\lceil \frac{n}{2^2} \right\rceil$, а след $(l-1)$ -то сравнение ще бъде $\left\lceil \frac{n}{2^{l-1}} \right\rceil$ т. е. един елемент. Следователно, необходимо е още най-много едно сравнение (т. е. общо най-много l), за да се намери търсеният елемент или да се установи, че такъв елемент няма. По

условие $1 \leq \frac{n}{2^{l-1}} < 2$, откъдето чрез логаритмуване следва: $0 \leq \log_2 n - \log_2 2^{l-1} < 1$, т. е. $l-1 \leq \log_2 n$.

Окончателно се получава $l \leq \lceil \log_2 n \rceil + 1$.

Да разгледаме отново алгоритъм I (подреждане чрез “вмъкване”). Точка I3 указва, че елементът a_{k+1} трябва да намери мястото си между първите k вече подредени елементи. С други думи, това е задача за търсене в нареден масив и следователно може да се използва алгоритъм **B**. Това е само един пример, който показва, че подреждането и търсенето са две взаимно свързани дейности. Ние подреждаме, за да търсим по-бързо, но като подреждаме, се налага да търсим. Във всички изложени дотук алгоритми за подреждане и търсене се предполагаше, че елементите в масива са наредени един след друг. Ако структурата на масива е друга, естествено тогава и алгоритмите за подреждане и търсене ще бъдат други, отчитащи характерните особености на съответната структура.

Блок-схема на алгоритъма:



В тази схема е използван знакът $\div$, означаващ деление на цели числа, в резултат на което се получава цяло число, ненадминаващо частното на двете числа. Например $3 \div 2 = 1$.

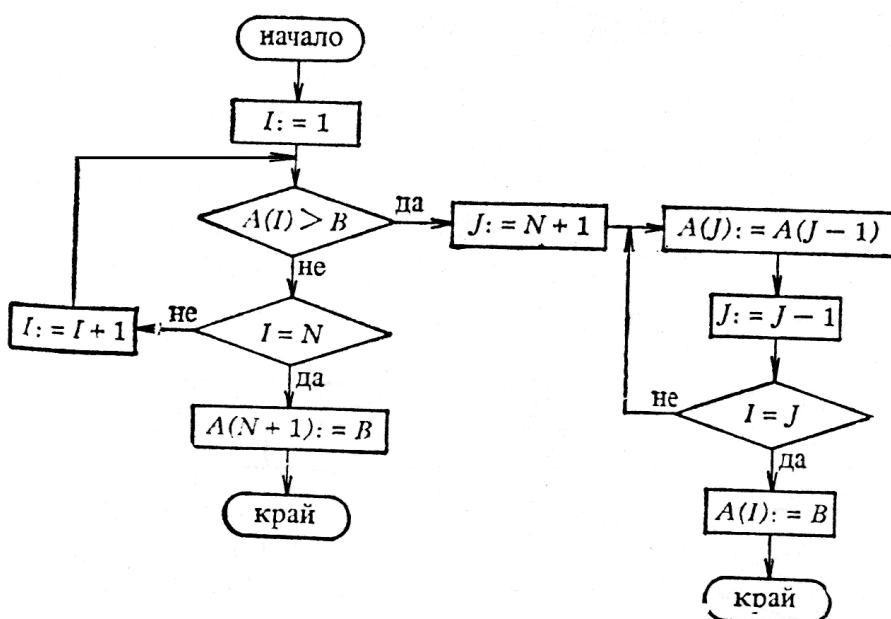
Търсене на определен елемент в подреден масив от данни често се налага в практиката. Примери за такава дейност са: търсене на картонче в библиотечен каталог, търсене на име в списък, подреден по азбучен ред; търсене на фактура в архив, нареден по датите на издаването, и др.

Задачата за търсене на елемент може да се съпровожда с извличане на някаква допълнителна информация за търсения елемент. В примера с таблицата на студентите може да е необходимо да се узнаят името и успехът на студент със зададен факултетен номер. Най-общо действията от такъв характер се наричат „извличане на справки“.

3. Вмъкване на елемент в подреден масив

Даден е масив A с N ($N \geq 2$) числени елемента, подредени по големина, и число B . Числото B да стане елемент на масива A така, че да се запази наредбата. Предполагаме, че сред елементите няма еднакви.

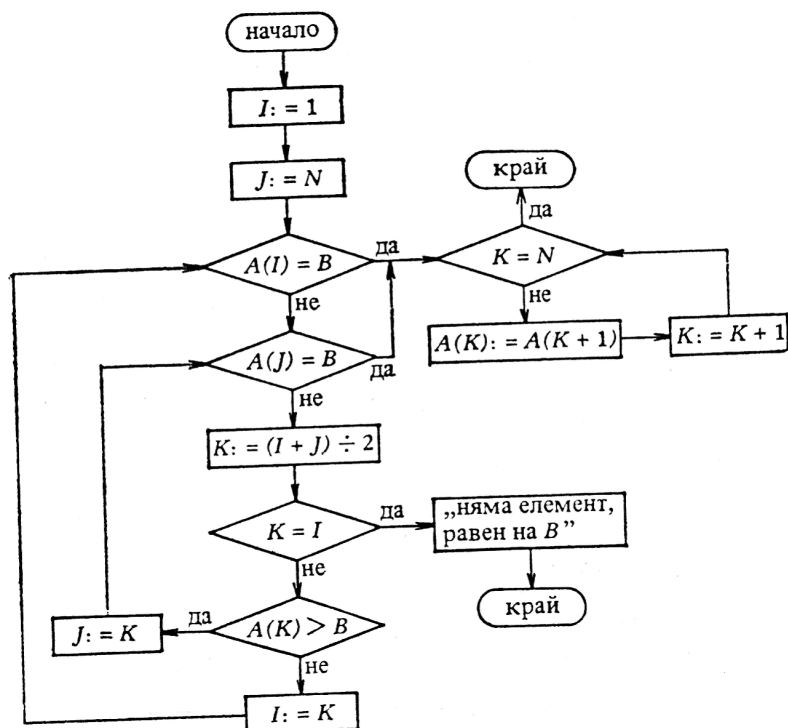
Блок-схема на алгоритъма:



4. Премахване на елемент от подреден масив

Даден е масив A с N ($N \geq 2$) числени елемента подредени по големина. Дадено е и числото B . Елементът на A , равен по стойност на B , да се премахне от масива A .

Следва блок-схема на алгоритъма.



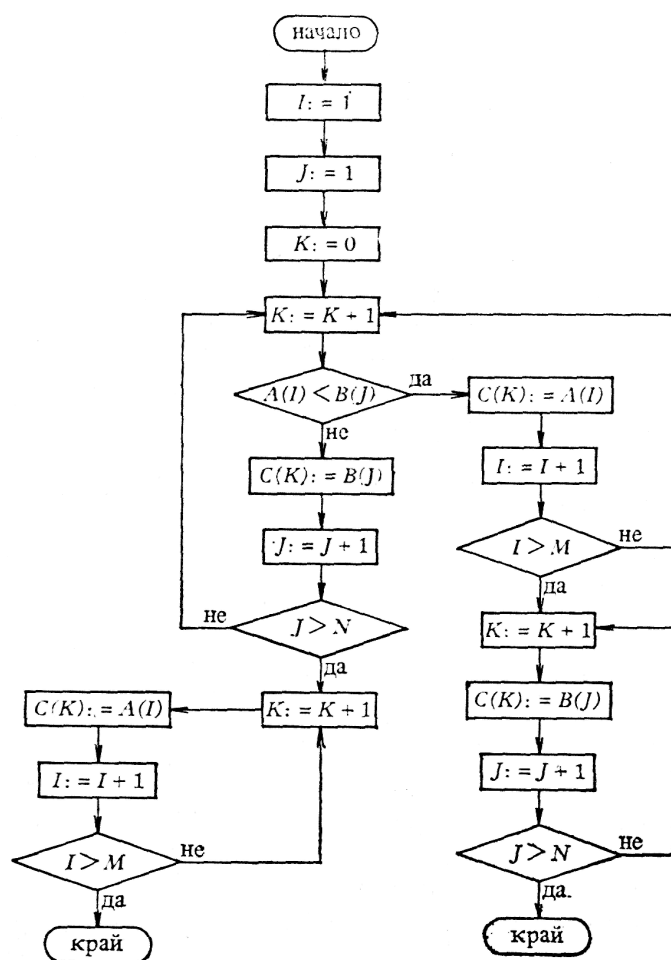
Най-общо действията, свързани с промени в масиви или таблици, се наричат актуализация, или обновяване. Те са характерни за изменящи се информационни системи.

Задача, която е типична при работа с информационни масиви.

Дадени са два подредени масива A и B съответно с по M и N числени елемента. Да се състави нов масив C който съдържа елементите на A и B и също да е подреден.

Ще предполагаме, че A и B нямат общи елементи. Задачи от този тип се наричат “задачи за сливане”.

Алгоритъм. Изразява се в следното: сравняват се по големина първият елемент на A с първия елемент на B . По-малкият от двата се записва като първи елемент на масива C . Следващото сравнение става отново над първите елементи, като считаме, че записаният елемент в C вече отсъства от съответния масив. Така се продължава, докато се изчерпи единият масив. Останалите елементи или елемент на другия масив се дописват в края на масива C . Блок-схема на алгоритъма:



[предишна страница](#)